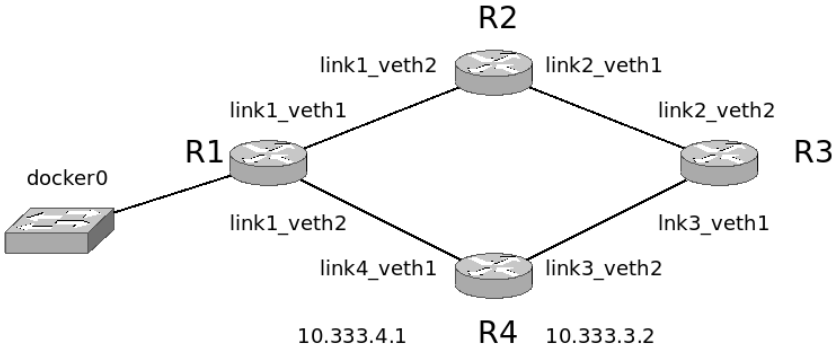
Encaminament Dinàmic bàsic amb RIP

Què farem:

- Crearem imatges docker amb debian-slim i part dels paquets **quagga**
- Crearem la infraestructura de xarxa enllaços tipus *veth*.
- Configurarem el protocol d'encaminament dinàmic interior **RIPv2**.
- Provarem caigudes de rutes i comprovarem com el RIP troba les alternatives existents, adaptant-se automàticament tot canviant les taules d'encaminament.

Topologia i esquema d'adreçament

Tindrem una topologia circular[3] amb dues rutes possibles entre qualsevol router:



L'esquema d'adreçament que usarem serà: **10.3B.#link.0/30**, on ...

- 3B és el tercer byte que teniu assignat a la DMZ de la pràctica 1. Al diagrama hi ha l'exemple de les IPs del R4 si la vostra assignació fos 198.18.333.0/24.
- El número de link (#link) es correspon amb el número més baix dels dos routers connectats a enllaç.

D'aquesta manera reconeixerem bé els links i a qui pertanyen les IPs.

Observació: Per a les xarxes dels links punt a punt és habitual usar la màscara /30, la qual permet dues adreces de host, que són les úniques que necessitem.

El contenidor R1 serà l'únic que es connecti a una network del docker, és a dir, que usi el pont **bridge0** per a donar accés als altres cap a Internet. A la resta de contenidors/routers **no usarem les docker networks** per a tenir un major control[4]. Per a connectar-los usarem enllaços virtuals tipus veth, a mode d'enllaços punt a punt.

3 *Reflexió: com és circular, els datagrames podrien fer voltes eternament ?*
4 *no volem més virtual bridges (amb IPs i gateways controlats pel docker)*

1. La infraestructura

Fareu un *script* anomenat **fes_docker_prac7.sh** que estableixi la topologia de la xarxa per a després poder fer les proves de connectivitat i resiliència.

1.1 Imatge docker

Creeu el fitxer **dockerfile_gsx_prac7** amb:

```
FROM debian:buster-slim
MAINTAINER Professor GSX (posa aquí el teu nom, no em suplantis, caram!)
# etc ....
ADD  ./prac7_config_rip.sh  /root
CMD [ "/bin/bash" ]
```

Necessitareu almenys els paquets: initscripts, quagga-ripd, iputils-ping i traceroute,
El contingut del fitxer **prac7_config_rip.sh** es detalla més endavant. De moment el creem buit.

1.2 Els contenidors

A excepció del R1, els *docker run* s'han de fer sense networks, doncs després crearem les veths dels links:

```
docker run $OPCIONS --hostname router$node --network=none --name R$node $imatge
```

Tot seguit crearem els enllaços, però per assignar-los necessitarem tenir els contenidors en marxa.

1.3 Els enllaços

Per aquesta part escriurem un script a banda que serà executat amb **sudo**, doncs per a crear els enllaços i assignar-los als contenidors haurem de tenir privilegis elevats[5].

El següent exemple mostra com crear un enllaç punt a punt virtual:

```
      ip link add link${node}_veth1 type veth peer name link${node}_veth2
```

Observació: podeu veure les noves interfícies a l'amfitrió amb ip link.

Ara que ho tenim tot creat podem assignar els enllaços als contenidors pertinents. Per exemple:

```
      pid=$(docker inspect --format '{{.State.Pid}}' R$node)
      ip link set netns $pid dev link${node}_veth1
```

Observació: com ara les veth pertanyen al namespace dels contenidors ja no les podrem veure des de l'amfitrió.

5 *totes les comandes amb sudo les podríeu agrupar en un script a banda i avisar*

També assignarem les IPs. Per exemple:

```
      nsenter -t $pid -n ip addr add 10.0.$node.1/30 dev link${node}_veth1
      nsenter -t $pid -n ip link set dev link${node}_veth1 up
```

1.4 Eliminar la infraestructura

Us pot ser útil un script que elimini els contenidors (les veth ja s'eliminaran).

2. Configuració dels routers

Heu d'escriure un script anomenat **prac7_config_rip.sh** que configuri els servei zebra[6] i el servei ripd. Per a configurar el primer simplement creeu el fitxer buit **/etc/quagga/zebra.conf**. La configuració bàsica del segon és prou senzilla. Exemple de configuració al fitxer **/etc/quagga/ripd,conf**:

```
router rip
  version 2
! comentari: falten les línies específiques pel R1
  network 10.0.0,0/30
  network .....
```

Usem la versió 2 perquè aquesta permet treballar amb *classless* IPs. Per a cada xarxa que vulgueu anunciar haurem d'afegir una línia network. Altrament les PDUs de RIP no s'enviarien en aquells enllaços.

En el cas de R1 haurem de fer que anuncia als altres routers la sortida cap a Internet. També haurem d'evitar que rebi informació RIP des de l'enllaç de sortida. Per això afegirem a les anteriors les següents clàusules:

```
  default-information originate
  passive-interface eth0
```

Un cop creat el fitxer de configuració haurem d'ajustar els permisos:

```
chown -R quagga.quaggavty /etc/quagga/
chmod 640 /etc/quagga/*.conf
```

i re-engegar els dos serveis.

Al R1 a més haureu d'afegir una mascarada per a que encamini el tràfic intern cap a Internet. D'aquesta manera podríem instal·lar no software a qualsevol contenidor.

Finalment, ara que els links estan assignats i les IPs també, ja podrem executar la configuració del protocol RIP.

```
docker exec R$node /root/prac7_config_rip.sh
```

6 *zebra is an IP routing manager. It provides kernel routing table updates, interface lookups, and redistribution of routes between different routing protocols.*

3. Proves

Tria un router que no sigui el R1. Comprovem que els dos *daemons* s'estan executant:

```
docker top $ROUTER
```

Comprovem la taula d'encaminament s'ha actualitzat amb informació de **zebra**:

```
docker exec $ROUTER ip -c route | tee -a sortida_prac7.txt
```

Quaaga emula la interfície de configuració de Cisco[7] una *shell* virtual (vtysh). Comprovem la configuració i la taula d'encaminament, la qual està més ben detallada:

```
docker exec $ROUTER vtysh | tee -a sortida_prac7.txt
show running config
show ip route
show ip rip status
exit
```

Pareu atenció als temps i a les distàncies.

Comprovem les IPs i les rutes cap a $ROUTER:

```
docker exec $ROUTER ip -c address
```

Tria una de les IPs, comprovem que s'hi arriba i comprovem la ruta:

```
docker exec R1 ping -c1 $IPdestiROUTER
docker exec R1 traceroute -n $IPdestiROUTER  | tee -a sortida_prac7.txt
docker exec $ROUTER traceroute -n -T $IPdestiROUTER  | tee -a sortida_prac7.txt
```

Observació: convé usar la opció -n (numèric) o altrament el traceroute tarda moltíssim, doncs intenta resoldre les IPs a nom (tot i que els RTT mostrats són els reals, és a dir, molt baixos).

Forceu la caiguda d'un enllaç d'aquesta ruta[8]:

```
docker exec .... ip link set l .... down | tee -a sortida_prac7.txt
docker exec R1 ip route get $IPdestiROUTER
```

Espereu a que la ruta no sigui la per defecte, és a dir, no sigui cap a Internet.

```
docker exec R1 traceroute -n $IPdestiROUTER | tee -a sortida_prac7.txt
```

Si voleu podeu aixecar de nou la interfície baixada administrativament, esperar uns segons i tronar a provar. S'usarà la ruta nova o la ruta anterior ? .... en realitat ara mateix no importa, n'hi ha prou amb tenir connectivitat.

7 *segurament la vàreu usar al programa packettracer a l'assignatura Xarxes de Dades*

8 *a tenir en compta: en un enllaç punt a punt quan cau un extrem també cau l'altre.*

Apunt final: Aquests protocols obtenen la informació de la topologia (que sol ser un graf cíclic bidireccional), apliquen un algoritme de camins mínims (com ara Bellman-Ford o Dijkstra) i generen un graf sense cicles, és a dir un arbre. En conseqüència, cada router deix d'usar alguns enllaços del graf per a evitar els cicles. Convé notar que cada router té el seu propi arbre i, per tant, se solen usar tots els enllaços.

Lliurament

Un tgz anomenat **prac7_$COGNOMS.tgz** amb:

- tots els fitxers necessaris per a fer la pràctica, sense cap subdirectori.
- el fitxer de les proves **sortida_prac7.txt**

Referències bàsiques

- man dockerfile, docker-exec, ...
- man ip-link, network_namespaces, nsenter

Referències extra

- www.nongnu.org/quagga/docs/docs-multi/RIP.html
- wikipedia.org/wiki/Babel_(protocol)

Curiositats

El projecte zebra ara està extingit i el seu successor és el project quagga. Les quagga con una subespècie extingida de les zebres però que s'està recuperant. El successor de quagga és FRRouting Project. A la seva web diu:

"Join the ranks of network architects using FRR for ISPs, SaaS infrastructure, web 2.0 businesses, hyperscale services, and Fortune 500 private clouds."